# Ejercicios Java 1 . Enlace [Repo Ejercios 1 java](https://github.com/cesarbouzas/2206_Bootcamp/tree/master/002_IDEs/001_Eclipse)

## Ejercicio 1: ejecutar un HolaMundo en java para comprobar el correcto funcionamiento del IDE y JDK .

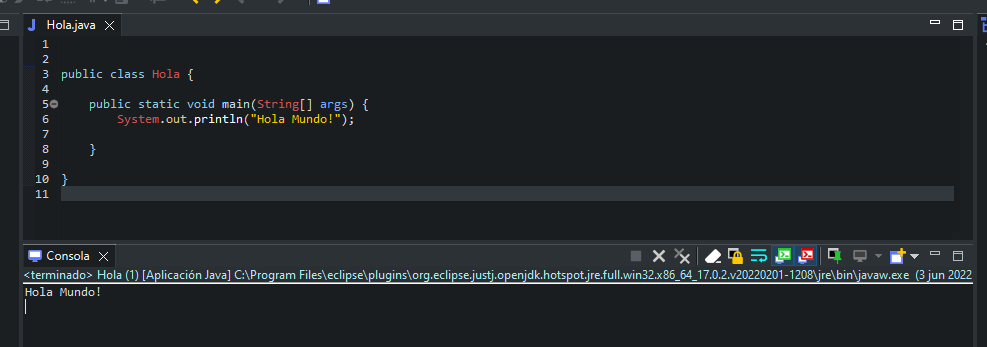


Ilustración 1 Hola Mundo!!!!

## Ejercicio 2: Dado el radio, (15), calcular el área del círculo y la longitud de la circunferencia.

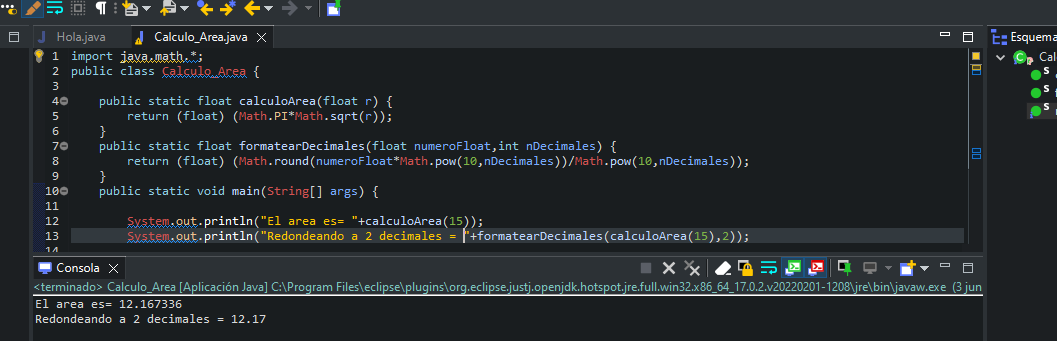


Ilustración 2 Calcular el área de un Circulo

## Ejercicio 3:

### Crear un método que reciba un nombre y salude a dicha persona.

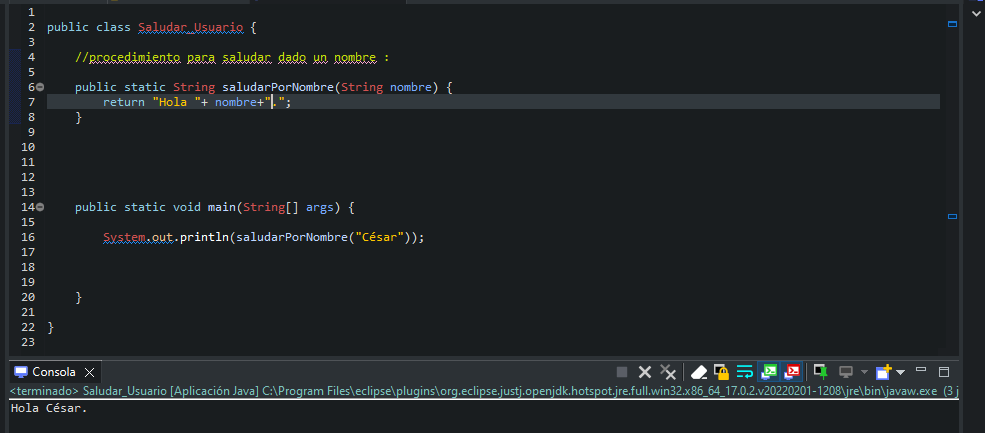


Ilustración 3 Saludar a César

### Crear un método con un mensaje de error (“usuario no encontrado”).

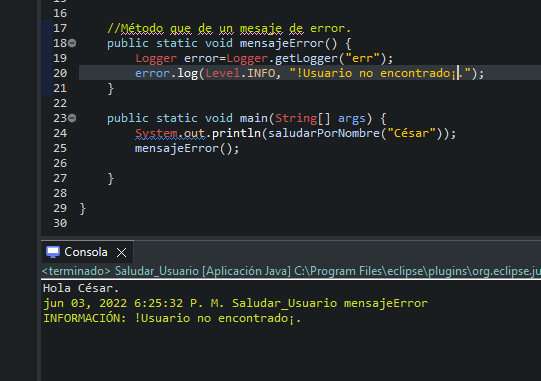


Ilustración mensaje de error

### Crear un procedimiento que reciba en nombre de una persona y verifique si es un usuario valido o no (crear con una constante un usuario valido ficticio).

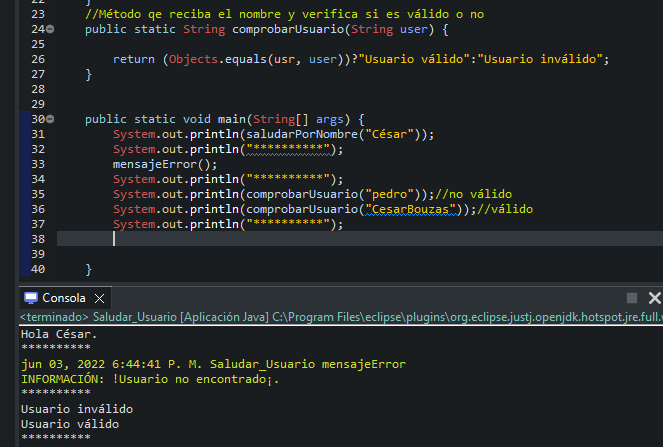


Ilustración control de usuario

### Hacer pruebas introduciendo distintos nombres de usuarios (podéis ajustarlo más introduciendo nombre y password, por ejemplo).

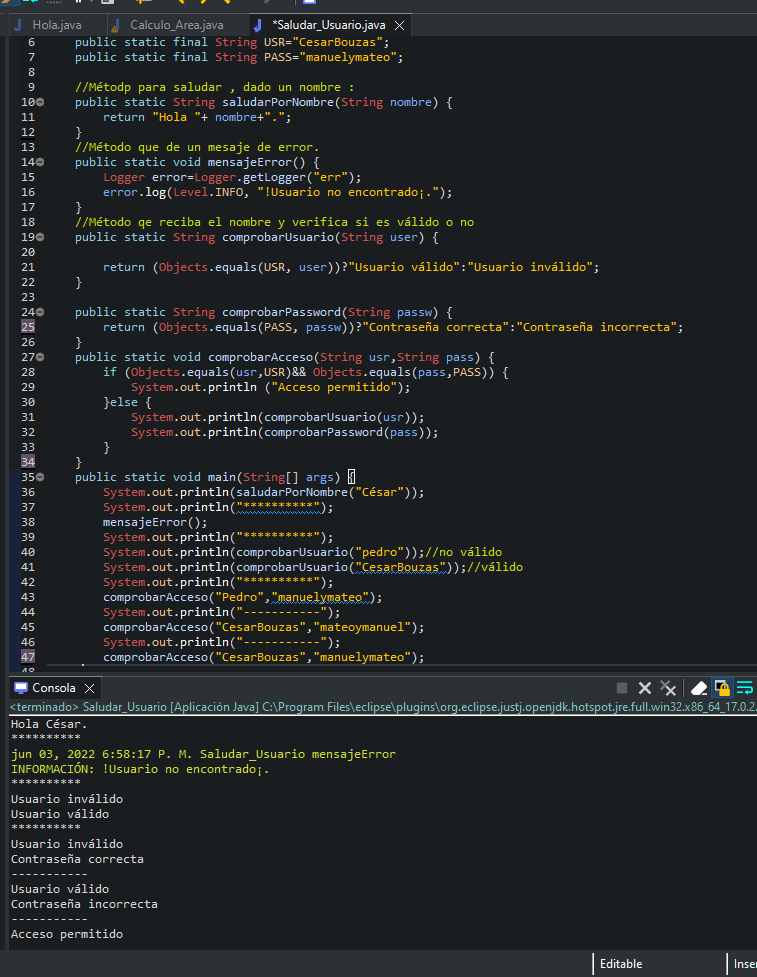


Ilustración control de usurio y password

## Ejercicio 4: Ejercicio para que creen ejemplos sin estructuras de control

### Suma los 5 primeros números.

System.out.println(1+2+3+4+5);

### Suma los 20 primeros números pares.

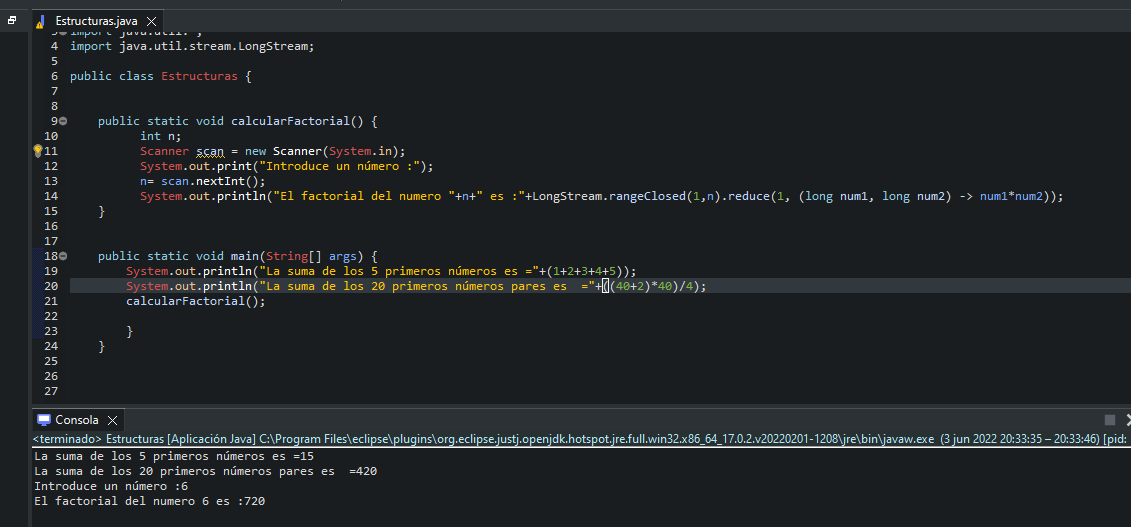


Ilustración sin Econtrol suma 5 primeros numero pares.

### Calcula el factorial del número que introduzcas.



Ilustración SinEcontrol factorial de un número

### Hacer lo mismo usando estructuras de control.

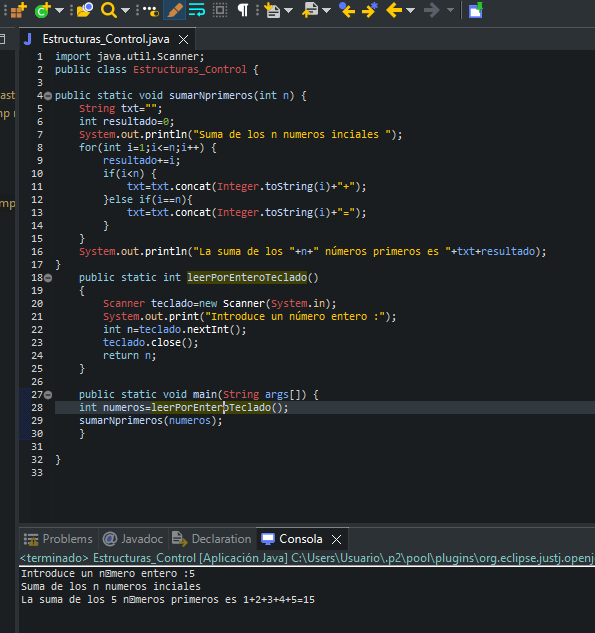


Ilustración suma de los n números primeros

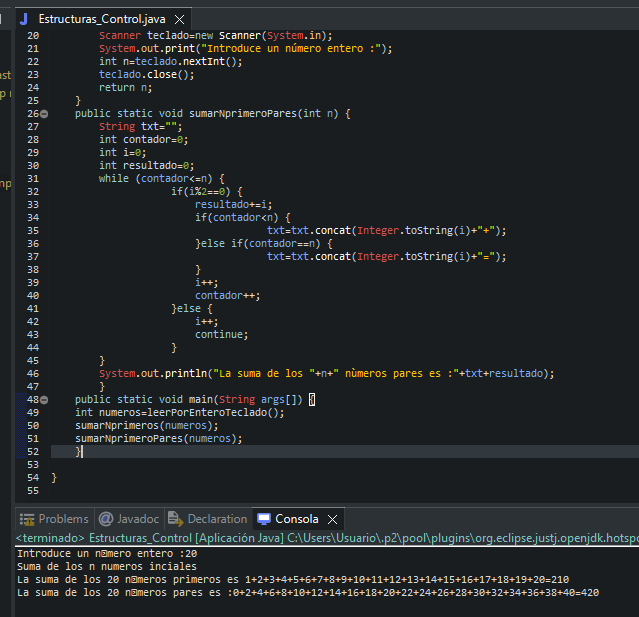


Ilustración suma de los N primeros pares

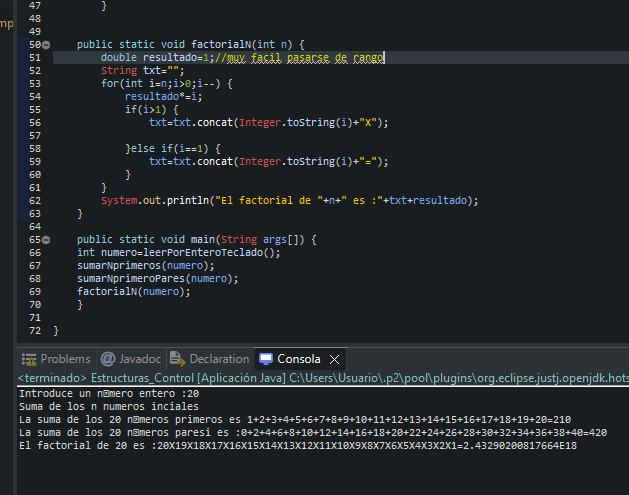


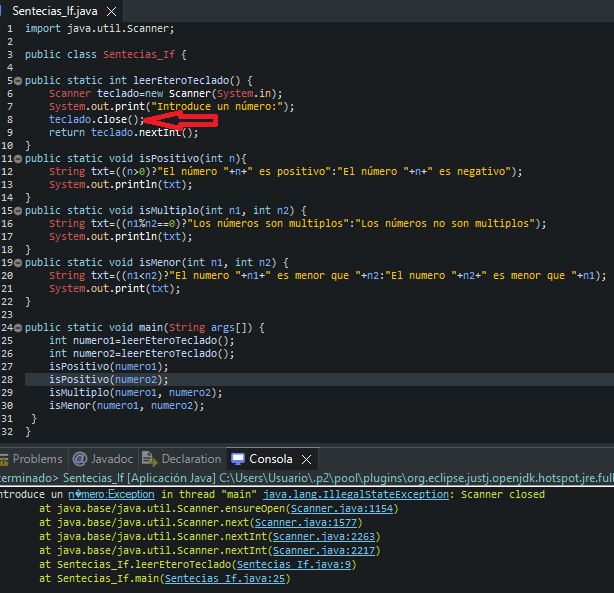
Ilustración Factorial de un número.

## Ejercicio 5: Crear ejercicios de ejemplo para usar las estructuras de control (sentencias if) Comprobar si un número es positivo o negativo,comprobar si un número es múltiplo de otro comprobar que un número es menor a otro.

### 

Ilustración multiplos , mayor y negativo

ERROR CON SCANNER CLOSE



## Ejercicio 6: Crear ejercicios de ejemplo para usar las estructuras de control (sentencia switch case) .

Este paquete java.time incluye muchas clases, pero las básicas son:

**LocalDate:** representa a fechas sin la hora y nos facilita su manejo para declararlas, sumar y restar fechas y compararlas.

**LocalTime:** es idéntica a la anterior pero para el manejo de horas, sin ninguna fecha asociada, pudiendo así compararlas, sumar o restar tiempo a las mismas...

**LocalDateTime:** como puedes suponer, es una combinación de las dos anteriores, que permite hacer lo mismo con fechas y horas simultáneamente.

**Instant**: es muy parecida a la anterior pero a la vez muy diferente. Se usa para almacenar un punto determinado en el tiempo, o sea con fecha y hora, pero guarda su valor como un timestamp de UNIX, es decir, en nanosegundos desde el epoch de UNIX (1/1/1970 a las 00:00) y usando la zona horaria UTC. Es muy útil para manejar momentos en el tiempo de manera neutra e intercambiarlo entre aplicaciones y sistemas, por lo que lo verás utilizado muy a menudo.

**ZonedDateTime:** esta clase es como la LocalDateTime pero teniendo en cuenta una zona horaria concreta, ya que las anteriores no la tienen en cuenta.

**Period**: esta clase auxiliar nos ayuda a obtener diferencias entre fechas en distintos periodos (segundos, minutos, días...) y también a añadir esas diferencias a las fechas.

**Duration**: esta es muy parecida a la anterior pero para manejo de horas exclusivamente..

Construyendo fechas y horas con java.time

Estas clases producen instancias inmutables, al contrario de lo que pasaba con las antiguas clases Date de Java, por lo que son thread-safe. Dado que carecen de constructores públicos, se instancian usando métodos de tipo "factoría", es decir, tienen métodos que construyen estas clases a partir de posibles parámetros que le pasemos.

En concreto, todas las de manejo de fechas y horas disponen de tres métodos importantes, que son:

**now(): crean instancias nuevas a partir de la fecha y hora actual.**

System.out.println("La fecha actual es: " + LocalDate.now());

System.out.println( "La hora actual es: " + LocalTime.now() );

System.out.println( "La fecha y hora actuales son: " + LocalDateTime.now() );

System.out.println( "El instante actual es: " + Instant.now() );

System.out.println( "La fecha y hora actuales con zona horaria son: " + ZonedDateTime.now() );

of(): construyen fechas y horas a partir de sus partes.

with(): modifican la fecha u hora actual en función del parámetro que se le pase, con alguna cantidad (años, días, horas...) o alguna clase de ajuste.

### Comprobar cuál es la estación del año (se pueden hacer por meses… Enero, Febrero,Marzo es INVIERNO, por ejemplo) .

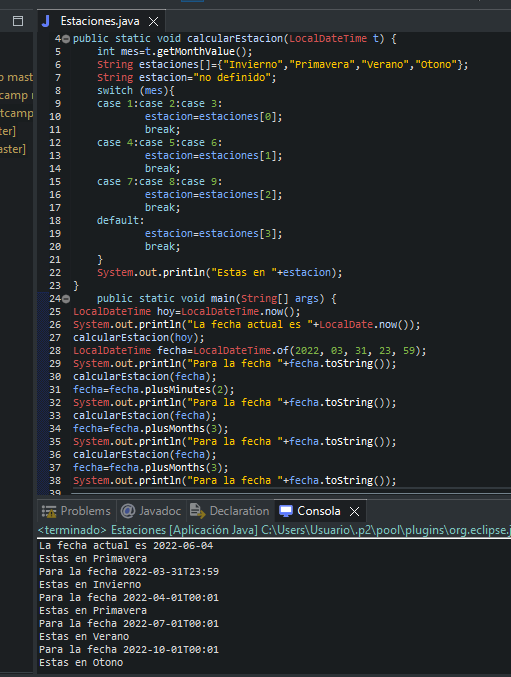


Ilustración Cálculo de estaciones.

### Comprobar si es de mañana tarde o noche :

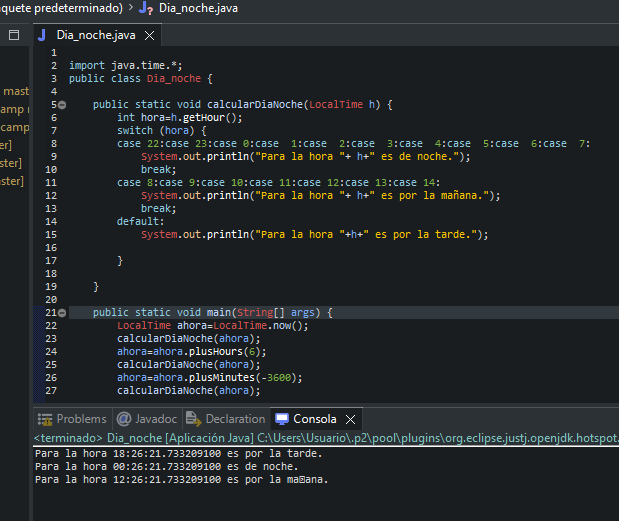


Ilustración cálculo de día , tarde o noche

## Ejercicio 7: estructuras de control:

### Mostrar los primeros 15 números naturales.

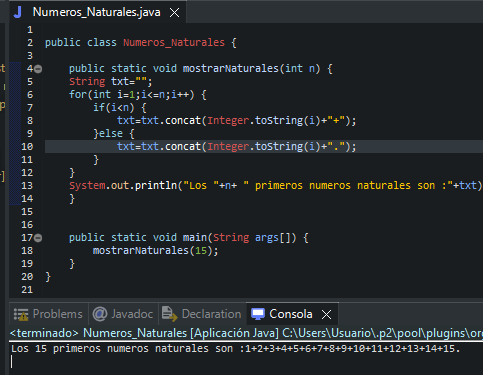


Ilustración n primeros naturales

### Sumar los primeros 100 números naturales.

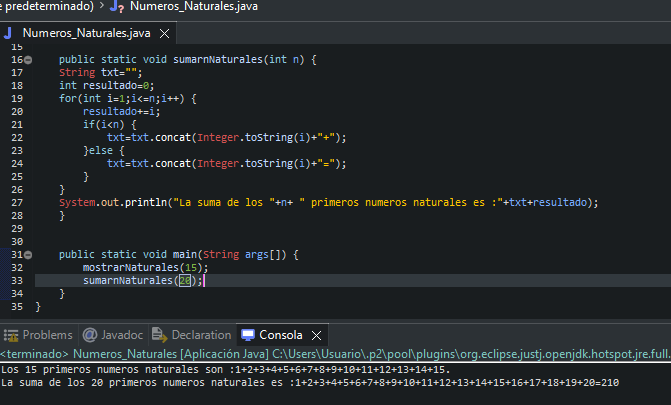


Ilustración suma de los n números naturales

### Buscar la posición de un elemento concreto en un array .

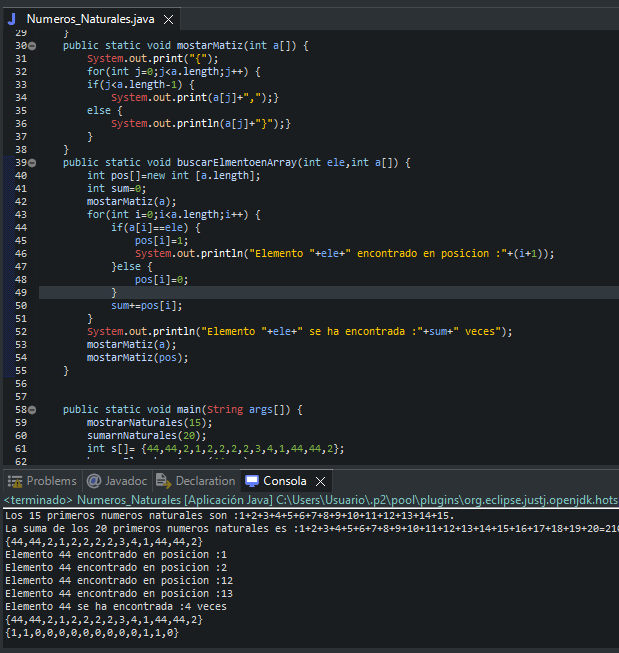


Ilustración Búsqueda en un Array

## Ejercicio 8: estructuras de control. Realizar un bucle mientras una variable incrementa su valor hasta un tope (usar while) .

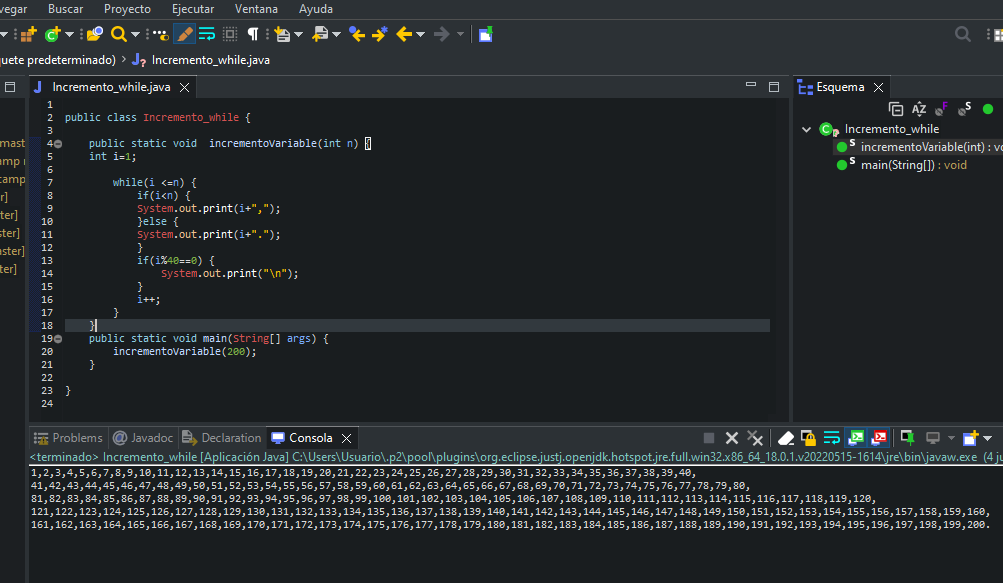


Ilustración lista while

## Ejercicio 9:  Realizar un bucle mientras una variable incrementa su valor hasta un tope (usar do while) .

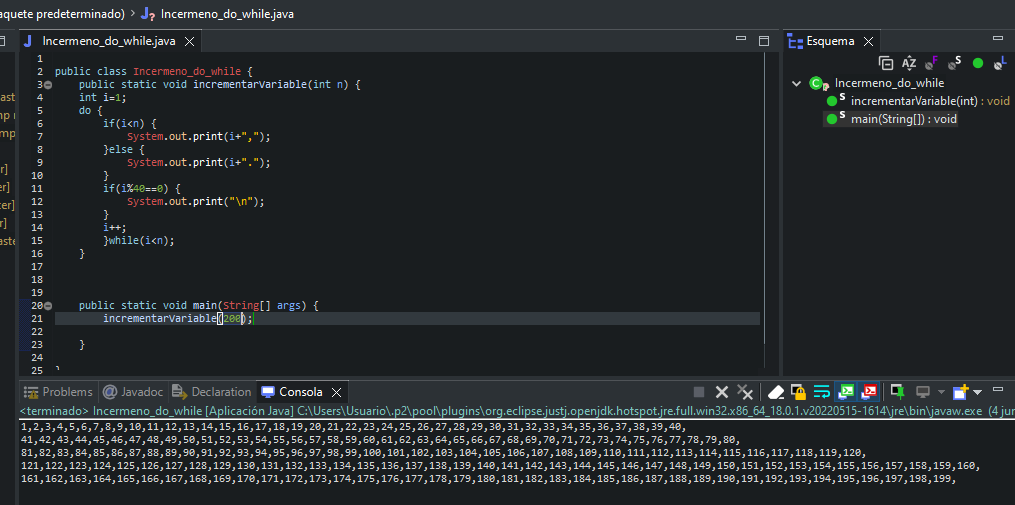


Ilustración do-while

## Ejercicio 10:

## Sacar 10 bolas de color aleatoriamente (azul, rojas y verdes).

## 

Ilustración Sacar Bolas Aleatroias

## Contar cuantas bolas azules salen en total.

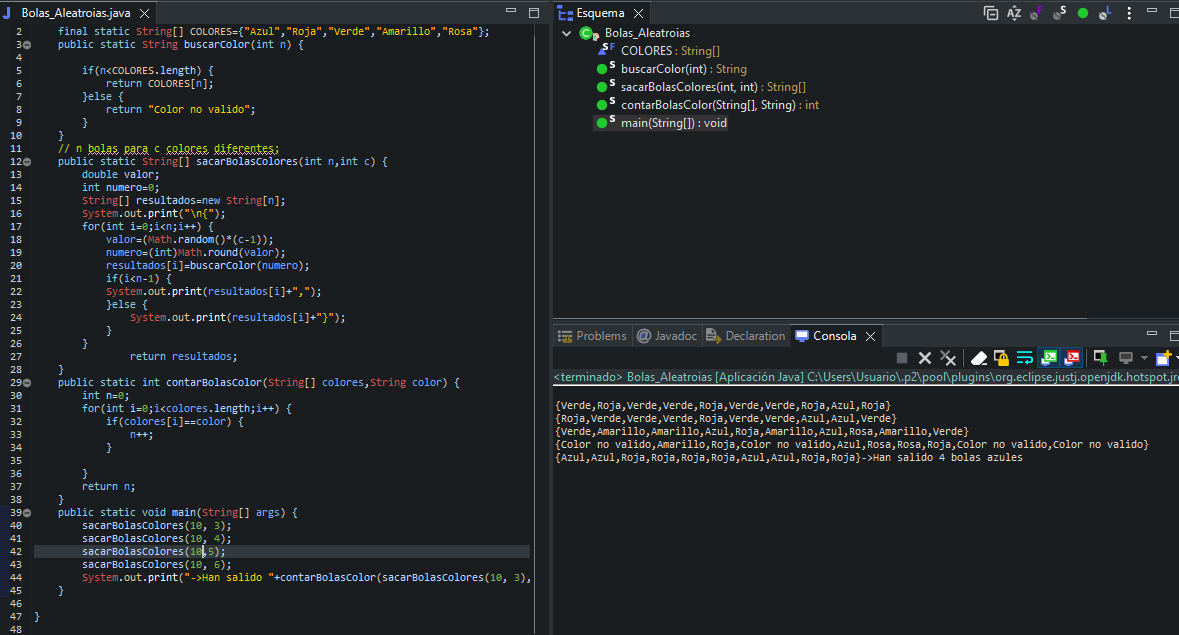


Ilustración contar bolas de color Azul

## Mostrar qué color sale hasta que aparezca la primera azul (incluida la azul).

